

# Approximate Partial Order Reduction

Zhenqi Huang and Sayan Mitra

University of Illinois at Urbana-Champaign

**Abstract.** We develop a partial order reduction method for labeled transition systems over metric spaces. We introduce the notion of  $\varepsilon$ -independent actions such that executing these actions in any order results in states that are close to each other. Then we define  $\varepsilon$ -equivalent action sequences that swap  $\varepsilon$ -independent action pairs. We present an algorithm to over-approximate the reach set of executions that take  $\varepsilon$ -equivalent action sequences. We are also able to show that the over-approximation can be computed up to arbitrary precision.

## 1 Introduction

We focus on systems modeled by discrete time models or *labeled transition systems* (LTS). In this model, nodes (or processes) take actions to communicate with each other and to update the local and shared states of the system. In an asynchronous network, the nodes runs concurrently and the actions may interleave arbitrarily. Thus, the number of the possible action sequences can be extremely large. We study the reach set computation problem for labeled transition systems. Computing the reach set for such systems is challenging mainly for two reasons. First, the set of initial states is uncountable and the number of executions (or trajectories) is uncountable. Hence explicit examination of every single execution is intractable. Second, the number of action sequences, even from a single initial state, is large owing to the combinatorial explosion arising from concurrency. In some cases, the number grows exponentially with the time bound. In this paper, we propose a verification method to address both challenges which exploits the sensitivity of executions to the ordering of actions.

Consider an asynchronous network consisting of  $N$  components. In the case where each component takes an action independent of others, the entire network can take  $N!$  different action sequences or executions. Checking the invariance property of the network then requires us to examine all these possible action sequences. Partial order reduction methods were first introduced in the context of model checking concurrent softwares to reduce the number of action sequences needed to be explored in performing model checking [18,26]. It exploits the fact that some pairs of actions taken by different nodes may be commutative, and therefore the executions obtained by swapping them need not to be explored both times. Precisely, two actions  $a$  and  $b$  are said to be *independent* if from any state, the resulting state would be the same regardless of the order in which  $a$  and  $b$  are executed as illustrated in Figure 1a. This independence relation allows one to define relation over the set of executions which swaps independent actions. Equivalent executions from the same initial state reach the same final state. If we can show that the invariance properties are indifferent to equivalent executions, then

checking invariance for a representative action sequence is sufficient for proving it for the whole equivalent class.

Partial order reduction has proven to be a powerful tool in software verification. It has been successfully applied to a variety of distributed systems, including leader election protocol [2], indexers [17], file systems [8], security protocol [7] and distributed schedulers [3]. In [8], a file transfer protocol with millions of states and transitions is verified in a matter of seconds using a partial order reduction method.

The conventional partial order methods have two limitations that hinder their application to cyber-physical systems. First, a pair of actions are considered independent only if they lead to *exactly* the same state regardless of the order of their execution. Such exactly independent action pairs are rare in CPS context, where the individual nodes often interact with a shared environment in the presence of noise and disturbances. This makes the partial order reduction methods ignore action pairs which lead to *nearly* but not exactly identical states. Second, there is no way to reason about executions with different initial states, even if they experience the same action sequences.

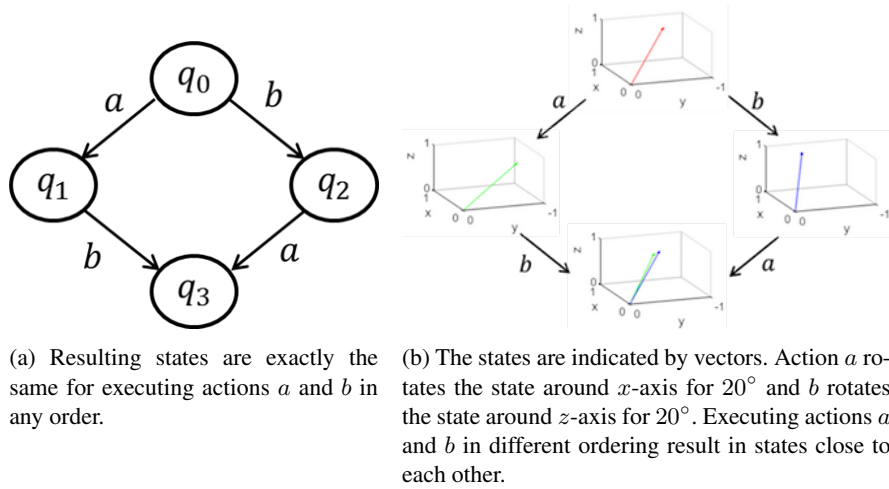


Fig. 1: Exactly and approximately independent actions

In this work, we develop partial order reduction techniques that take advantage of information about the sensitivity of the transitions of the individual agents in the system to prune executions that are approximately equivalent. First of all, we assume that there is a metric on the state space. We introduce the notion of *approximate independence* of actions. Specifically, with a parameter  $\varepsilon \geq 0$  chosen by the user, two actions  $a$  and  $b$  are  $\varepsilon$ -independent if from any state, the resulting states are within  $\varepsilon$  distance regardless of the order in which actions  $a$  and  $b$  are executed. An example of approximate independent actions is illustrated in Figure 1b. Smaller of the parameter  $\varepsilon$ , the final states after executing  $\varepsilon$ -independent actions in any order are closer to each other. The conventional (exact) independence relation, as illustrated in Figure 1a, is a special case of

$\varepsilon$ -independence relation with  $\varepsilon = 0$ . With this extension, we can construct equivalent classes of action sequences by swapping approximately independent actions. Although executions that follow equivalent action sequences may not necessarily reach the same final state, the final states should be close to each other.

Together with this approximate independence relation, we exploit the continuity of the transition functions for approximating reachable states. Roughly, the continuity of an action  $a$  captures the notion that executing action  $a$  from two states that are close to each other, should result in states that remain close. Using the continuity property, we are able to inductively compute the distance between a pair of executions using the distance between their initial states. Combining continuity with  $\varepsilon$ -independence relation, though careful analysis which will be presented in Section 6 we can quantify the distance between executions starting from initial states that are close to each other and follow equivalent action sequences.

In a nutshell, partial order reduction is model checking using representatives from equivalent classes of executions [9,25]. The reduction is more efficient if a larger size of equivalent class can be represented by a particular execution. Now with the notions of  $\varepsilon$ -independent actions and continuity, we expand the representation power of a single execution. We will show examples in Section 8 that for models where the conventional partial order reduction does not apply, our method can efficiently compute an over-approximation of states reached by a class of executions close to a given representative.

The rest of this paper is organized as follows. First in Section 2 we discuss related works. In Section 3, we introduce a modeling framework for infinite state systems and a notion of continuity of actions. Then in Section 4, we introduce the notion of  $\varepsilon$ -independent actions and equivalent classes of action sequences. In Sections 5-7, we gradually develop an algorithm to compute over-approximated reach set of all equivalent executions using a reduced set of action sequences. This algorithm is applied to verify invariance properties of a linear transition system and a heater control system in Section 8.

## 2 Related works

There are two main classes of partial order methods are proposed in last few decades: persistent set methods and sleep set methods. For each state of the system, the *persistent/ample set* methods compute a subset of enabled transitions—the persistent set (or ample set)—such that the omitted transitions are independent to those selected [9,2]. The reduced system which only considers the transitions in the persistent set is guaranteed to represent all behaviors of the original system. The persistent sets and the reduced systems are often derived by static analysis of the code. More recently, researchers developed the *sleep set* methods to avoid the static analysis [30,1,17]. These methods examine the history of actions taken by an execution and decide a set of actions that need to be explored in the future. The set of omitted actions is the sleep set. The persistent set and sleep set methods can be used complementary [17].

In the context of cyber-physical systems, robustness analysis are developed to quantify the closeness of executions with disturbances. The authors of [21] presented an algorithm to compute the output deviation with bounded disturbance. The algorithm com-

bines symbolic execution and optimization techniques. In [6], the authors introduce the robustness analysis to many classic algorithms and proposed a automated framework to prove robustness for softwares. Later in [28], this technique is extended to verify robustness of networked systems.

### 3 Background

We denote by  $len(\tau) \in \mathbb{N}$  the length of  $\tau$ . For any for  $i \in [len(\tau)]$ ,  $\tau(i)$  is the  $i$ -th action in  $\tau$ . The state of our labeled transition system is defined by the valuations of a set of finite-valued or real-valued variables. Each variable  $v$  has a type,  $type(v)$ , which is either the set of reals or some finite set. For a set of variables  $V$ , a valuation  $\mathbf{v}$  maps each  $v \in V$  to a point in  $type(v)$ . The set of all valuations of  $V$  is  $Val(V)$ .

#### 3.1 Transition systems

**Definition 1.** A Labeled Transition System  $\mathcal{A}$  is a tuple  $\langle X \cup L, \Theta, A, \rightarrow \rangle$  where (i)  $X$  is a set of real-valued variables and  $L$  is a set of finite-valued variables.  $Q = Val(X \cup L)$  is the set of states, (ii)  $\Theta \subseteq Q$  is a compact set of initial states, (iii)  $A$  is a finite set of actions, and (iv)  $\rightarrow \subseteq Q \times A \times Q$  is a transition relation.

A state  $q \in Q$  is a valuation of the real-valued and finite-valued variables. We denote by  $q.X$  and  $q.L$ , respectively, the continuous (real-valued) and discrete (finite-valued) part of the state  $q$ . We will view the continuous part  $q.X$  as a vector in  $\mathbb{R}^{|X|}$  by fixing an arbitrary ordering of  $X$ . The norm  $|\cdot|$  on  $q.X$  is an arbitrary norm; wherever we use specific norms we will make it explicit. For  $\delta \geq 0$ , the  $\delta$ -neighborhood of  $q$  is denoted by  $\mathcal{B}_\delta(q) \triangleq \{q' \in Q : q'.L = q.L \wedge |q'.X - q.X| \leq \delta\}$ .

For any  $(q, a, q') \in \rightarrow$ , we write  $q \xrightarrow{a} q'$ . For any action  $a \in A$ , its guard is the set  $guard(a) = \{q \in Q \mid \exists q' \in Q, q \xrightarrow{a} q'\}$ . An action  $a$  is *deterministic* if for any state  $q \in Q$ , if there exists  $q_1, q_2 \in Q$  with  $q \xrightarrow{a} q_1$  and  $q \xrightarrow{a} q_2$ , then  $q_1 = q_2$ .

*Assumption.* We assume actions are deterministic and specified by a guard and a *transition function* and we overload the name of an action  $a$  with its transition function. Thus, for each  $q \in guard(a)$ ,  $q \xrightarrow{a} a(q)$ . We further assume that for any state pair  $q, q'$ , if  $q.L = q'.L$  then  $a(q).L = a(q').L$ .

*Executions and traces.* For a deterministic transition system, a state  $q_0 \in Q$  and a finite action sequence  $\tau = a_0 a_1 \dots a_{n-1}$  uniquely specifies a *potential execution*  $\xi_{q_0, \tau} = q_0, a_0, q_1, a_1, \dots, a_{n-1}, q_n$  where for each  $i \in [n]$ ,  $a_i(q_i) = q_{i+1}$ . An *execution* is a potential execution with (i)  $q_0 \in \Theta$  and (ii) for each  $i \in [n]$ ,  $q_i \in guard(a_i)$ . For any potential execution  $\xi_{q_0, \tau}$ , its trace is the action sequence  $\tau$ , that is  $trace(\xi_{q_0, \tau}) = \tau \in A^*$ . The length of  $\xi_{q_0, \tau}$  is defined as the length of its trace and  $\xi_{q_0, \tau}(i) = q_i$  is the state visited after the  $i$ -th transition. The first and last state on a execution  $\xi$  are denoted as  $\xi.fstate = \xi(0)$  and  $\xi.lstate = \xi(len(\xi))$ .

For a subset of initial states  $S \subseteq \Theta$  and a time bound  $T \geq 0$ , we denote by  $Execs(S, T)$  as the set of length  $T$  executions of the system from initial states  $S$ .

We denote by  $\text{Traces}(S, T) \triangleq \{\text{trace}(\xi) \mid \xi \in \text{Execs}(S, T)\}$  as the set of length  $T$  traces that are taken by executions from  $S$ . We denote the *reach set at time  $T$*  by  $\text{Reach}(S, T) \triangleq \{\xi.\text{lstate} \mid \xi \in \text{Execs}(S, T)\}$ . Our goal is to over-approximate  $\text{Reach}(\Theta, T)$  exploiting partial order reduction.

1	<b>automata</b> Consensus( $n \in \mathbb{N}, N \in \mathbb{N}$ )	<b>transitions</b>	1
	<b>variables</b>	<b>a<sub>i</sub> for each <math>i \in [N]</math></b>	
3	$x : \mathbb{R}^n$	<b>pre</b> $\neg d_i$	3
	$d : \mathbb{B}^N$	<b>eff</b> $x := A_i x \wedge d_i := \text{true}$	
5	<b>initially</b>	<b>a<sub>⊥</sub></b>	5
	$x_i \in [-4, 4]$ <b>for each</b> $i \in [n]$	<b>pre</b> $\bigwedge_{i \in [N]} d_i$	
7	$d_i := \text{false}$ <b>for each</b> $i \in [n]$	<b>eff</b> $d_i := \text{false}$ <b>for each</b> $i \in [N]$	7

Fig. 2: Labeled transition system model of iterative consensus.

*Example 1 (Iterative Consensus).* A range of distributed and concurrent systems can be modeled in this framework, for example, consensus, clock synchronization [29,27], and distributed control applications [16,22]. An iterative consensus protocol with  $N$  processes is shown in Figure 2. The state of the system is a vector  $x$  in  $\mathbb{R}^n$  and each process  $i$  changes the state by performing the linear transformation  $x \leftarrow A_i x$ . The system evolves in rounds: in each round, each process  $i$  exactly once but in arbitrary order. The boolean-valued vector  $d$  is used to track of the processes that have acted in a round. Many iterative consensus, flocking, and coverage algorithms have this structure [5,15,20,24,23]; the *exactly one time per round* requirement gives fairness and can be relaxed.

The set of actions is  $A = \{a_i\}_{i \in [N]} \cup \{a_{\perp}\}$ . For each  $i \in [N]$ , the action  $a_i$  is enabled when  $d_i$  is *false* and when it occurs  $x$  is updated as  $A_i x$ , where  $A_i$  is an  $n \times n$  matrix. The action  $a_{\perp}$  can occur only when all  $d_i$ 's are set to *true* and it resets all the  $d_i$ 's to *false*. For an instance with  $N = 3$ , a valid execution could have the trace  $\tau = a_0 a_2 a_1 a_{\perp} a_1 a_0 a_2 a_{\perp}$ .

It can be checked that the assumptions made at the beginning of Section 3 are satisfied: Each action is deterministic. For any  $q, q' \in Q$  and  $a_i \in A$ , if  $q.d = q'.d$ , action  $a_i$  simply sets  $d_i$  to *true* and keeps all other components unchanged, and hence,  $a_i(q).d = a_i(q').d$ . Action  $a_{\perp}$  resets all  $d_i$ , hence  $a_{\perp}(q).d = a_{\perp}(q').d$ . In fact, the assumption will continue to hold if  $A_i x$  is replaced by a nonlinear transition function  $a_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$ .

### 3.2 Discrepancy functions

A discrepancy function bounds the changes in a system's executions as a continuous function of the changes in its inputs or initial states. In [11,19,13,10] several methods for computing discrepancy have been introduced and it has been shown hoe they can be used for approximating the reachable sets of continuous time models. Here, we extend the notion of discrepancy to labeled transition systems. A discrepancy for an action

bounds the changes in the continuous state brought about by the action's transition function.

**Definition 2.** For an action  $a \in A$ , a continuous function  $\beta_a : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$  is a discrepancy function if for any pair of states  $q, q' \in Q$  with  $q.L = q'.L$ ,

- (i)  $|a(q).X - a(q').X| \leq \beta_a(|q.X - q'.X|)$ , and
- (ii)  $\beta_a(\cdot) \rightarrow 0$  as  $|q.X - q'.X| \rightarrow 0$ .

The first property gives an upper-bound on the changes brought about by action  $a$  and the second property ensures that the bound given by  $\beta_a$  can be made arbitrarily small by choosing continuous states that are close. As the following proposition states, given discrepancy functions for actions, we can reason about distance between executions that share the same trace but have different initial states.

**Proposition 1.** Suppose each action  $a \in A$  has a discrepancy function  $\beta_a$ . For any  $T \geq 0$  and action sequence  $\tau = a_0 a_1 a_2 \dots a_T$ , and for any pair of states  $q, q' \in Q$  with  $q.L = q'.L$ , the last states of the pair of potential executions satisfy:

$$\xi_{q,\tau}.lstate.L = \xi_{q',\tau}.lstate.L, \quad (1)$$

$$|\xi_{q,\tau}.lstate.X - \xi_{q',\tau}.lstate.X| \leq \beta_{a_T} \beta_{a_{T-1}} \dots \beta_{a_0} (|q.X - q'.X|). \quad (2)$$

*Example 2.* Consider an instance of Consensus of Example 1 with  $n = 3$  and  $N = 3$  with the standard 2-norm on  $\mathbb{R}^3$ . Let the matrices  $A_i$  be

$$A_0 = \begin{bmatrix} 0.2 & -0.2 & -0.3 \\ -0.2 & 0.2 & -0.1 \\ -0.3 & -0.1 & 0.3 \end{bmatrix}, A_1 = \begin{bmatrix} 0.2 & 0.3 & 0.2 \\ 0.3 & -0.2 & 0.3 \\ 0.2 & 0.3 & 0 \end{bmatrix}, A_2 = \begin{bmatrix} -0.1 & 0 & 0.4 \\ 0 & 0.4 & -0.2 \\ 0.4 & -0.2 & -0.1 \end{bmatrix}.$$

All the matrices are stable, and it can be checked that for any state pair  $q, q' \in Q$  with  $q.L = q'.L$ ,

$$|a_i(q).X - a_i(q').X|_2 \leq |A_i|_2 |q.X - q'.X|_2.$$

Where the induced 2-norms of the matrices are  $|A_0|_2 = 0.57, |A_1|_2 = 0.56, |A_2|_2 = 0.53$ . Thus, for any  $v \in \mathbb{R}_{\geq 0}$ , the discrepancy functions of  $a_0, a_1, a_2$  can be defined as:

$$\beta_{a_0}(v) = 0.57v, \beta_{a_1}(v) = 0.56v, \text{ and } \beta_{a_2}(v) = 0.53v.$$

For actions with nonlinear transition functions, computing global discrepancy functions is difficult in general but local approaches [14] can be adequate for computing reachable sets from compact initial sets.

## 4 Independent actions and neighboring executions

Central to partial order methods is the notion of independent actions. A pair of actions are said to be independent if from any starting state the resulting states are exactly the same regardless of the order in which they are executed (see Figure 1a). We extend this notion with an approximation parameter  $\varepsilon \geq 0$ , and define a pair of actions to be  $\varepsilon$ -independent if the resulting states are within  $\varepsilon$  distance if their ordering is swapped.

#### 4.1 Approximately independent actions

**Definition 3.** For  $\varepsilon \geq 0$ , two distinct actions  $a, b \in A$  are  $\varepsilon$ -independent, denoted by  $a \stackrel{\varepsilon}{\sim} b$ , if for any state  $q \in Q$

- (i) (Commutativity)  $ab(q).L = ba(q).L$ , and
- (ii) (Closeness)  $|ab(q).X - ba(q).X| \leq \varepsilon$ .

This definition extends the standard definition of independence (see e.g. Definition 8.3 of [4]) in two ways. First, it allows the continuous states to mismatch by a little  $\varepsilon > 0$  in two executions in which  $\varepsilon$ -independent actions are swapped. Therefore, the enabledness of future actions may change. Suppose actions  $a$  and  $b$  are  $\varepsilon$ -independent with  $\varepsilon = 0$ , then as illustrated in Figure 3a action  $c$  is enabled at state  $q_2$  independent of which path is taken to arrive at  $q_2$ . With  $\varepsilon > 0$  (Figure 3b), swapping actions  $a$  and  $b$  makes the execution deviate from the state  $q_2$  to  $q'_2$  and  $c$  may not be enabled at  $q'_2$ . If  $\xi_{q_0,abc}$  is an execution, then the potential execution  $\xi_{q_0,bac}$  may not be an execution. Later we will compute a ball around an execution  $\xi_{q_0,abc}$  which will be guaranteed to contain all potential executions derived by swapping  $\varepsilon$ -independent action pairs of. In Section 6, we will show that for small enough  $\varepsilon$ , this side effect can be made negligible and the reach set can be over-approximated precisely.

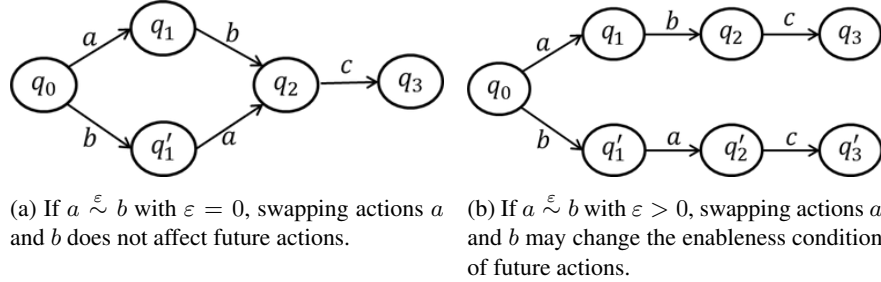


Fig. 3: The enabledness condition guarantees that swapping independent actions in an execution results in a valid execution. However, the same property does not hold for approximately independent actions.

The second extension we made in Definition 3 is that action  $b$  need not be enabled at state  $a(q)$ . As a consequence, for a pair of  $\varepsilon$ -independent actions  $a$  and  $b$ , if  $\xi_{q_0,ab}$  is an execution,  $\xi_{q_0,ba}$  may not be one.

In the notion of  $\varepsilon$ -independence, the parameter  $\varepsilon \geq 0$  captures the degree of the approximation. For a smaller  $\varepsilon$ , the independent relation is more restrictive. For an action sequence  $\tau \in A^*$  and an action  $a \in A$ ,  $\tau$  is  $\varepsilon$ -independent to  $a$ , written as  $\tau \stackrel{\varepsilon}{\sim} a$ , if  $\tau$  is empty string or for every  $i \in [\text{len}(\tau)]$ ,  $\tau(i) \stackrel{\varepsilon}{\sim} a$ .

It is clear that the approximate independence relation over  $A$  is symmetric, but it is not transitive in general.

*Example 3.* Consider approximate independence of actions in Consensus. Fix any  $i, j \in [N]$  such that  $i \neq j$  and any state  $q \in Q$ . It can be checked that:

$$a_i a_j(q).d_k = a_j a_i(q).d_k = \begin{cases} \text{true}, & \text{if } k \in \{i, j\} \\ q.d_k, & \text{otherwise.} \end{cases}$$

Hence, we have  $a_i a_j(q).d = a_j a_i(q).d$  and the commutativity condition of Definition 3 holds. For the closeness condition, we have

$$|a_i a_j(q).x - a_j a_i(q).x|_2 = |(A_i A_j - A_j A_i)q.x|_2 \leq |A_i A_j - A_j A_i|_2 |q.x|_2. \quad (3)$$

If the matrices  $A_i$  and  $A_j$  commute, then  $a_i$  and  $a_j$  are  $\varepsilon$ -approximately independent with  $\varepsilon = 0$ . If  $Q$  is a compact set or the system has a bounded invariant set, then  $|q.X|_2$  is bounded and there exists a finite  $\varepsilon \geq 0$  such that  $a_i \stackrel{\varepsilon}{\sim} a_j$ . For the specific matrices presented in Example 2, the all stable. For any any state  $q$  and any  $i \in [3]$ , the change in the squared 2-norm is:

$$|a_i(q).x|_2^2 - |q.x|_2^2 = q.x^\top A_i^\top A_i q.x - q.x^\top q.x = q.x^\top (A_i^\top A_i - I_3) q.x.$$

It can be checked that, for each  $i \in [3]$ ,  $A_i^\top A_i - I_3$  is a negative definite matrix. Hence  $|a_i(q).x|_2 \leq |q.x|_2$ . That is, the norm of the continuous state is non-increasing. Suppose initially  $x \in [-4, 4]^3$  then the 2-norm of the initial state is bounded by the value  $4\sqrt{3}$ . Since the norm of state is non-increasing in any transitions,  $Inv = \{x \in \mathbb{R}^3 : |x|_2 \leq 4\sqrt{3}\}$  is an invariant of the system. From (3), we have  $|a_0 a_1(q).x - a_1 a_0(q).x|_2 \leq 0.1$ ,  $|a_0 a_2(q).x - a_2 a_0(q).x|_2 \leq 0.07$ , and  $|a_1 a_2(q).x - a_2 a_1(q).x|_2 \leq 0.17$ . Thus, with  $\varepsilon = 0.1$ , it follows that  $a_0 \stackrel{\varepsilon}{\sim} a_2$  and  $a_1 \stackrel{\varepsilon}{\sim} a_2$  and  $\stackrel{\varepsilon}{\sim}$  is not transitive, but with  $\varepsilon = 0.2$   $\stackrel{\varepsilon}{\sim}$  is transitive.

Definition 3 implies that from any state  $q$ , executing two  $\varepsilon$ -independent actions in either order, we end up in states that are within  $\varepsilon$  distance. The following proposition bounds the distance between potential executions with different initial states using discrepancy.

**Proposition 2.** For a pair of  $\varepsilon$ -independent actions  $a, b \in A$ , two states  $q, q' \in Q$  with  $q.L = q'.L$ , (i)  $ba(q).L = ab(q').L$ , and (ii)

$$|ba(q).X - ab(q').X| \leq \beta_b \beta_a(|q.X - q'.X|) + \varepsilon,$$

where  $\beta_a, \beta_b$  are discrepancy functions of  $a, b$  respectively.

*Proof.* Fix a pair of states  $q, q' \in Q$  with  $q.L = q'.L$ . Since  $a \stackrel{\varepsilon}{\sim} b$ , we have  $ba(q).L = ab(q').L$ . Using the Assumption, we have  $ab(q).L = ab(q').L$ . Using triangular inequality, we have  $|ba(q).X - ab(q').X| \leq |ba(q).X - ba(q').X| + |ba(q').X - ab(q').X|$ . The first term is bounded by  $\beta_b \beta_a(|q.X - q'.X|)$  using the property of discrepancy functions and the second is bounded by  $\varepsilon$  by the definition of approximate independence, and hence, the result follows.



## 4.2 Equivalent action sequences

For a set of  $\varepsilon$ -independent actions, we define an equivalence relation on the space of finite action sequences  $A^*$ .

**Definition 4.** For any  $\varepsilon \geq 0$ , we define a relation  $R \subseteq A^* \times A^*$  such that  $\tau R \tau'$  iff there exists  $\sigma, \eta \in A^*$  and  $a, b \in A$  such that

$$a \stackrel{\varepsilon}{\sim} b, \tau = \sigma ab \eta, \text{ and } \tau' = \sigma ba \eta.$$

We define an equivalence relation  $\stackrel{\varepsilon}{\equiv} \subseteq A^* \times A^*$  called  $\varepsilon$ -equivalence, as the reflexive and transitive closure of  $R$ .

That is, a pair of traces  $\tau, \tau' \in A^*$  is  $\varepsilon$ -equivalent if we can construct  $\tau'$  from  $\tau$  by swapping consecutive  $\varepsilon$ -independent actions. In the following proposition, we show that two potential executions with the same initial discrete state (location) and equivalent trace share the same final locations.

**Proposition 3.** Fix potential executions  $\xi = \xi_{q_0, \tau}$  and  $\xi' = \xi_{q'_0, \tau'}$ . If  $\xi$  and  $\xi'$  have the same initial location and equivalent traces, then they have the same final location. That is, if  $q_0.L = q'_0.L$  and  $\tau \stackrel{\varepsilon}{\equiv} \tau'$ , then  $\xi.Lstate.L = \xi'.Lstate.L$ .

*Proof.* If  $\tau = \tau'$ , then the proposition follows from the Assumption. Suppose  $\tau \neq \tau'$ , from Definition 4, there exists a sequence of action sequences  $\tau_0, \tau_1, \dots, \tau_k$  to join  $\tau$  and  $\tau'$  by swapping neighboring approximately independent actions. Precisely the sequence  $\{\tau_i\}_{i=0}^k$  satisfies: (i)  $\tau_0 = \tau$  and  $\tau_k = \tau'$ , and (ii) for each pair  $\tau_i$  and  $\tau_{i+1}$ , there exists  $\sigma, \eta \in A^*$  and  $a, b \in A$  such that  $a \stackrel{\varepsilon}{\sim} b$ ,  $\tau_i = \sigma ab \eta$ , and  $\tau_{i+1} = \sigma ba \eta$ . From Definition 3, swapping approximately independent actions preserves the value of the discrete part of the final state. Hence for any  $i \in [k]$ ,  $\xi_{q_0, \tau_i}.Lstate.L = \xi_{q_0, \tau_{i+1}}.Lstate.L$ . Therefore,  $\xi.Lstate.L = \xi'.Lstate.L$ .

The next definition captures pairs of potential executions that take equivalent action sequences and have initial states that are close together. In Lemma 2 we will show that such executions remain close to each other, which will be the basis for reachability analysis.

**Definition 5.** For  $\delta, \varepsilon \geq 0$ , a pair of potential executions  $\xi = \xi_{q_0, \tau}$  and  $\xi' = \xi_{q'_0, \tau'}$  are  $(\delta, \varepsilon)$ -close, denoted by  $\xi \stackrel{\delta, \varepsilon}{\approx} \xi'$ , if  $q_0.L = q'_0.L$ ,  $|q_0.X - q'_0.X| \leq \delta$ , and  $\tau \stackrel{\varepsilon}{\equiv} \tau'$ .

It follows from Proposition 3 that the final locations of any pair of  $(\delta, \varepsilon)$ -close potential executions are the same. In Section 6, we quantify the distance between  $(\delta, \varepsilon)$ -close potential executions. Then, by bloating a single potential execution, we can over-approximate the reach set of all potential executions close to it.

*Example 4.* In Example 3, we show that  $a_0 \stackrel{\varepsilon}{\sim} a_2$  and  $a_1 \stackrel{\varepsilon}{\sim} a_2$  with  $\varepsilon = 0.1$ . Consider the executions

$$\xi = q_0, a_2, q_1, a_1, q_2, a_0, q_3, a_{\perp}, q_4, \text{ and } \xi' = q'_0, a_1, q'_1, a_0, q'_2, a_2, q'_3, a_{\perp}, q'_4.$$

with traces  $trace(\xi) = a_2 a_1 a_0 a_{\perp}$  and  $trace(\xi') = a_1 a_0 a_2 a_{\perp}$ . For  $\varepsilon = 0.1$ , we have  $a_2 a_1 a_0 a_{\perp} \stackrel{\varepsilon}{\equiv} a_1 a_2 a_0 a_{\perp}$  and  $a_1 a_2 a_0 a_{\perp} \stackrel{\varepsilon}{\equiv} a_1 a_0 a_2 a_{\perp}$ . Since the equivalence relation  $\stackrel{\varepsilon}{\equiv}$  is transitive, we have  $trace(\xi) \stackrel{\varepsilon}{\equiv} trace(\xi')$ . Suppose  $q_0 \in \mathcal{B}_{\delta}(q'_0)$ , then  $\xi$  and  $\xi'$  are  $(\delta, \varepsilon)$ -close executions with  $\varepsilon = 0.1$ .

## 5 Interleaving Independent Actions

In this section, we present several results that parallel the standard development of partial order reduction in model checking. First, we bound the distance between potential executions that are derived by inserting a single action to an execution. Building up on this result, we develop an inductive method for computing distance between any execution and its  $(\delta, \varepsilon)$ -close neighbors. Throughout this section, we consider actions that are mutually  $\varepsilon$ -independent. In Section 6, we generalize these result to apply approximate partial order reduction to reachability with general actions.

We begin with few definitions. For a finite set of discrepancy functions  $\{\beta_a\}_{a \in S}$  corresponding to a set of actions  $S \subseteq A$ , we define  $\beta_{max} = \max_{a \in S} \{\beta_a\}$  as the point-wise upper bound of discrepancy functions. From Definition 2, for each  $a \in S$ ,  $\beta_a(|q.X - q'.X|) \rightarrow 0$  as  $|q.X - q'.X| \rightarrow 0$ . Hence, as the maximum of  $\beta_a$ , we have  $\beta_{max}(|q.X - q'.X|) \rightarrow 0$  as  $|q.X - q'.X| \rightarrow 0$ . It can be checked that  $\beta_{max}$  is a discrepancy function of each  $a \in S$ .

For an  $n \geq 0$  and a function  $\beta_{max}$  defined as above, we define a function  $\gamma_n = \sum_{i=0}^n \beta_{max}^i$ ; here  $\beta^i = \beta \beta^{i-1}$  for  $i \geq 1$  and  $\beta^0$  is the identity mapping. Using the properties of discrepancy functions as in Definition 2, we can show the following properties of  $\{\gamma_n\}_{n \in \mathbb{N}}$ .

**Proposition 4.** *Fix any finite subset of discrepancy functions  $\{\beta_a\}_{a \in S}$  with  $S \subseteq A$ . Let  $\beta_{max} = \max_{a \in S} \{\beta_a\}$  be the maximum function. For any  $n \geq 0$ ,  $\gamma_n = \sum_{i=0}^n \beta_{max}^i$  satisfies*

- (i) *for any  $\varepsilon \in \mathbb{R}_{\geq 0}$  and any  $n \geq n' \geq 0$ ,  $\gamma_n(\varepsilon) \geq \gamma_{n'}(\varepsilon)$ , and*
- (ii)  *$\lim_{\varepsilon \rightarrow 0} \gamma_n(\varepsilon) = 0$ .*

*Proof.* (i) For any  $n \geq 1$ , we have  $\gamma_n - \gamma_{n-1} = \beta_{max}^n$ . Since  $\beta_{max}^n = \max_{a \in S} \{\beta_a^n\}$  for some finite  $S$ , using Definition 2,  $\beta_{max}^n$  takes only non-negative values. Hence, the sequence of functions  $\{\gamma_n\}_{n \in \mathbb{R}_{\geq 0}}$  is non-decreasing.

(ii) Using the property of discrepancy functions, we have  $\lim_{\varepsilon \rightarrow 0} \beta_{max}(\varepsilon) = 0$ . By induction on the nested functions, we have  $\lim_{\varepsilon \rightarrow 0} \beta_{max}^i(0) = 0$  for any  $i \geq 0$ . Hence for any  $n \in \mathbb{R}_{\geq 0}$ ,  $\lim_{\varepsilon \rightarrow 0} \gamma_n(\varepsilon) = \lim_{\varepsilon \rightarrow 0} \sum_{i=0}^{n-1} \beta_{max}^i(\varepsilon) = 0$ .

### 5.1 Insertion of independent action

We will quantify the distance between two potential executions with equivalent traces, using the above  $\gamma_n$  function. Our first step involves computing the distance between two potential executions after inserting a single action at different positions. We study a simple scenario where the inserted action is independent to all others in the execution.

**Lemma 1.** *Consider any  $\varepsilon \geq 0$ , an initial state  $q_0 \in Q$ , an action  $a \in A$  and an action sequence  $\tau \in A^*$  with length  $n \geq 1$ . If  $a$  and  $\tau$  are  $\varepsilon$ -independent, then the potential executions  $\xi = \xi_{q_0, \tau a}$  and  $\xi' = \xi_{q_0, a \tau}$  satisfy*

- (i)  *$\xi'.lstate.L = \xi.lstate.L$ , and*
- (ii)  *$|\xi'.lstate.X - \xi.lstate.X| \leq \gamma_{n-1}(\varepsilon)$ , where  $\gamma_n$  corresponds to the set of discrepancy functions  $\{\beta_c\}_{c \in \tau}$  for the actions in  $\tau$ .*

*Proof.* Part (i) directly follows from Proposition 3. We will prove part (ii) by induction on the length of the action sequence  $\tau$ .

**Base:** For any action sequence  $\tau$  of length 1,  $\xi$  and  $\xi'$  are of the form  $\xi = q_0, b_0, q_1, a, q_2$  and  $\xi' = q_0, a, q'_1, b_0, q'_2$ . Since  $a \stackrel{\varepsilon}{\sim} b_0$  and the two executions start from the same state, it follows from Definition 3 that  $|q'_2.X - q_2.X| \leq \varepsilon$ .

Recall from the preliminary that  $\gamma_0(\varepsilon) = \beta^0(\varepsilon) = \varepsilon$ . Hence  $|q'_2.X - q_2.X| \leq \gamma_0(\varepsilon)$  holds for action sequence  $\tau$  with length  $n = 1$ .

**Induction:** Suppose the lemma holds for any  $\tau$  with length at most  $n - 1$ . Fixed any  $\tau = b_0 b_1 \dots b_{n-1}$  of length  $n$ , we will show the lemma holds for  $\tau$ . Let the potential executions  $\xi = \xi_{q_0, \tau a}$  and  $\xi' = \xi_{q_0, a \tau}$  be of the form

$$\begin{aligned}\xi &= q_0, b_0, q_1, b_1, \dots, b_{n-1}, q_n, a, q_{n+1}, \\ \xi' &= q_0, a, q'_1, b_0, q'_2, b_1, \dots, b_{n-1}, q'_{n+1}.\end{aligned}$$

It suffices to prove that  $|\xi.\text{lstate}.X - \xi'.\text{lstate}.X| = |q_{n+1}.X - q'_{n+1}.X| \leq \gamma_{n-1}(\varepsilon)$ . We first construct a potential execution  $\xi'' = \xi_{q_0, b_0 a b_1 \dots b_{n-1}}$  by swapping the first two actions of  $\xi'$ . Then,  $\xi''$  is of the form

$$\xi'' = q_0, b_0, q_1, a, q''_2, b_1, \dots, b_{n-1}, q''_{n+1}.$$

The potential executions  $\xi$ ,  $\xi'$  and  $\xi''$  are shown in Figure 4. We first compare the poten-

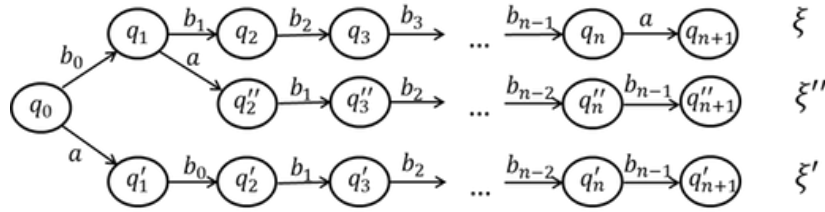


Fig. 4: Executions  $\xi = \xi_{q_0, \tau a}$ ,  $\xi' = \xi_{q_0, a \tau}$ , and  $\xi''$  which is constructed by swapping the first two actions in  $\xi'$ .

tial executions  $\xi$  and  $\xi''$ . Notice that,  $\xi$  and  $\xi''$  share a common prefix  $q_0, b_0, q_1$ . Starting from  $q_1$ , the action sequence of  $\xi''$  is derived from  $\text{trace}(\xi)$  by inserting action  $a$  in front of the action sequence  $\tau' = b_1 b_2 \dots b_{n-1}$ . Since  $\tau' \stackrel{\varepsilon}{\sim} a$ , applying the induction hypothesis on the length  $n - 1$  action sequence  $\tau'$ , we get

$$|q_{n+1}.X - q''_{n+1}.X| \leq \gamma_{n-2}(\varepsilon). \quad (4)$$

Then, we compare the potential executions  $\xi'$  and  $\xi''$ . Since  $b_0 \stackrel{\varepsilon}{\sim} a$ , by applying the property of Definition 3 to the first two actions of  $\xi'$  and  $\xi''$ , we have  $|q'_2.X - q''_2.X| \leq \varepsilon$ . We note that  $\xi'$  and  $\xi''$  have the same suffix of action sequence from  $q'_2$  and  $q''_2$ . Using Proposition 1 from states  $q'_2$  and  $q''_2$ , we have

$$|q'_{n+1}.X - q''_{n+1}.X| \leq \beta_{b_1} \beta_{b_2} \dots \beta_{b_{n-1}} (|q'_2.X - q''_2.X|) \leq \beta^{n-1}(\varepsilon). \quad (5)$$

Combining (4) and (5) with triangular inequality, we have

$$\begin{aligned} |q_{n+1}.X - q'_{n+1}.X| &\leq |q_{n+1}.X - q''_{n+1}.X| + |q'_{n+1}.X - q''_{n+1}.X| \\ &\leq \gamma_{n-2}(\varepsilon) + \beta^{n-1}(\varepsilon) = \gamma_{n-1}(\varepsilon). \end{aligned} \quad (6)$$

Therefore, the lemma holds for the action sequence  $\tau$  with length  $n$ , which completes the induction.

## 5.2 Permutation of independent actions

In Lemma 2, we analyzed distance between executions after inserting one single action. Using the lemma, we are going to compute an upper bound of the final distance between a potential execution  $\xi$  and its  $(\delta, \varepsilon)$ -close potential executions. We start with a definition of representative executions.

**Definition 6.** For any  $\delta, \varepsilon, r \in \mathbb{R}_{\geq 0}$  and any potential execution  $\xi = \xi_{q_0, \tau}$ ,  $\xi$  is a  $(\delta, \varepsilon, r)$ -representative potential execution if any potential execution  $\xi'$  that is  $(\delta, \varepsilon)$ -close to  $\xi$  satisfies

$$\xi'.\text{lstate} \in \mathcal{B}_r(\xi.\text{lstate}).$$

That is, for a  $(\delta, \varepsilon, r)$ -representative potential execution  $\xi$ , the  $r$ -neighborhood of its last state  $\mathcal{B}_r(\xi.\text{lstate})$  contains the last states of all its  $(\delta, \varepsilon)$ -close potential executions— $\xi$  can be used to represent or compute the reachsets of its neighbors. The parameter  $r$  is the *representative radius* of  $\xi$ . The following lemma gives a inductive way of constructing representative potential executions. Roughly, it shows how the representative radius  $r$  changes if an action  $a$  is appended to a potential execution  $\xi$ . First, we start with a special case where the appended action  $a$  is  $\varepsilon$ -independent of all other actions on  $\xi$ .

**Lemma 2.** Fix any  $\delta, \varepsilon, r \in \mathbb{R}_{\geq 0}$ , any  $(\delta, \varepsilon, r)$ -representative potential execution  $\xi_{q_0, \tau}$ , and any  $a \in A$  such that  $\tau \stackrel{\varepsilon}{\sim} a$ . Then,  $\xi = \xi_{q_0, \tau a}$  is a  $(\delta, \varepsilon, r')$ -representative potential execution with

$$r' = \beta_a(r) + \gamma_{\text{len}(\tau)-1}(\varepsilon). \quad (7)$$

*Proof.* Fix any  $\xi'$  such that  $\xi' \stackrel{\delta, \varepsilon}{\approx} \xi$  with initial state  $q'_0 \in \mathcal{B}_\delta(q_0)$ . It follows from Proposition 3 that  $\xi'.\text{lstate}.L = \xi.\text{lstate}.L$ . It suffices to prove that  $|\xi'.\text{lstate}.X - \xi.\text{lstate}.X| \leq r'$ .

Since  $\text{trace}(\xi') \stackrel{\varepsilon}{\equiv} \tau a$ ,  $\text{trace}(\xi')$  is in a form  $\phi a \eta$  with some  $\phi \eta \stackrel{\varepsilon}{\equiv} \tau$ . We construct a potential execution  $\xi'' = \xi_{q'_0, \phi \eta a}$ . The three potential executions are illustrated in Figure 5 below.

We note that the prefix  $(q_0, \tau, q_n)$  of  $\xi$  is a  $(\delta, \varepsilon, r)$ -representative potential execution. Since  $\phi \eta \stackrel{\varepsilon}{\equiv} \tau$  and  $q'_0 \in \mathcal{B}_\delta(q_0)$ , it follows from Definition 6 that  $|q_n.X - q''_n.X| \leq r$ . Hence

$$|\xi.\text{lstate}.X - \xi''.\text{lstate}.X| \leq \beta_a(|q_n.X - q''_n.X|) \leq \beta_a(r). \quad (8)$$

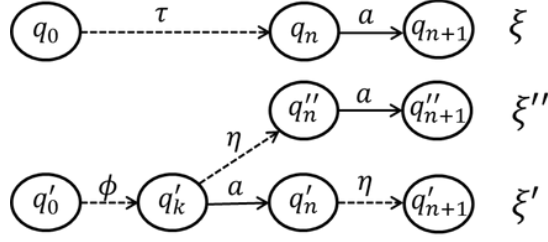


Fig. 5: Execution  $\xi$ , its  $\varepsilon$ -equivalent execution  $\xi'$ , and execution  $\xi''$  that is constructed by swapping action  $a$  to the back in  $\xi'$ .

On the other hand, we observe that the traces of  $\xi'$  and  $\xi''$  differ only in the position of action  $a$ . Application of Lemma 1 on  $\xi'$  and  $\xi''$  yields

$$|\xi'.\text{lstate}.X - \xi''.\text{lstate}.X| \leq \gamma_{\text{len}(\eta)-1}(\varepsilon) \leq \gamma_{\text{len}(\tau)-1}(\varepsilon). \quad (9)$$

Combining (8) and (9) with triangular inequality, we have

$$|\xi.\text{lstate}.X - \xi'.\text{lstate}.X| \leq \beta_a(r) + \gamma_{\text{len}(\tau)-1}(\varepsilon).$$

In Lemma 2, we analyze how the representative radius changes after appending a single action  $a$  to a potential execution  $\xi_{q_0, \tau}$ . Here we consider the special case where action  $a$  is  $\varepsilon$ -independent to the trace of  $\tau$ . In the next section, we generalize this results to any action  $a$  and trace  $\tau$ , without requirements on their independence condition.

## 6 Generalization of Executions

In this section, we will compute the representative radius  $\delta_T$  for a given execution  $\xi$  of length  $T$ , such that  $\xi$  is a  $(\delta_0, \varepsilon, \delta_T)$ -representative potential execution, for given parameters  $\delta_0, \varepsilon \geq 0$ . Our method involves computing a sequence  $\{\delta_t\}_{t=0}^T$  inductively such that  $\delta_t$  is the representative radius of the length  $t$  prefix of  $\xi$ .

Let action  $a$  be the  $t^{\text{th}}$  action on  $\xi$  and  $\tau$  be the length  $t$  prefix of  $\text{trace}(\xi)$ . If action  $a$  is  $\varepsilon$ -independent to  $\tau$ , then the representative radius  $\delta_t$  can be computed from  $\delta_{t-1}$  using Lemma 2. In this section, we generalize this result to the case where action  $a$  is not necessarily  $\varepsilon$ -independent to the whole sequence  $\tau$ . First, we will introduce the notion of *anchor position* in Definition 7 of  $a$  in  $\tau$ , which is the left most position of action  $a$  in all equivalent traces of  $\tau$ . Then, we will present Algorithm 2 for computing representative radius using anchor position.

### 6.1 Anchor position

In the proof of Lemma 2, we use the fact that any  $\varepsilon$ -equivalent trace of  $\tau a$  must be in a form of  $\phi a \eta$ . We observe that, the representative radius of  $\xi_{q_0, \tau a}$  depends on the length of  $\phi$  and  $\eta$ , as presented in Equation (9). For computing the representative radius, we

need the maximum length of  $\eta$ , or equivalently the minimum length of  $\phi$ . We introduce the notion of *anchor position* to capture this quantity.

For any sequence  $\tau \in A^*$  and an action  $a \in \tau$  in  $\tau$ , we define  $\tau.lastPos(a)$  as the largest index  $k$  such that  $\tau(k) = a$ . The *anchor position*,  $anchor(\tau, a)$  is the first possible position of  $a$  in any  $\tau'$  that is equivalent to  $\tau a$ . We formally define this quantity as follows.

**Definition 7.** For any action sequence  $\tau \in A^*$  and any action  $a \in A$ , the anchor position of  $a$  on  $\tau$  is

$$\min_{\tau' \stackrel{\varepsilon}{\equiv} \tau a} \tau'.lastPos(a).$$

For any trace  $\tau a$ , its  $\varepsilon$ -equivalent traces can be derived by swapping consecutive  $\varepsilon$ -independent action pairs. Hence, the anchor position of  $a$  is the leftmost position it can be swapped to, starting from the end. We note that any equivalent trace of  $\tau a$  can be written in a form  $\phi a \eta$  where  $\phi$  and  $\eta$  are the prefix and suffix of the last occurrence of action  $a$ . Hence, an equivalent way of defining the anchor position is:

$$anchor(\tau, a) = \min_{\phi a \eta \stackrel{\varepsilon}{\equiv} \tau a, a \notin \eta} len(\phi).$$

In the following algorithm, we find the anchor position of action  $a$  on an action sequence  $\tau$ . For any trace  $\tau$  and action  $a$ ,  $anchor(\tau, a)$  constructs a trace  $\phi \in A^*$ . Initially  $\phi$  is set to be the empty sequence. Iteratively, from the end of  $\tau$ , we add action  $\tau(t)$  to  $\phi$  if it is not independent to the entire trace  $\phi a$ . We will prove that, length of  $\phi$  gives the anchor position of action  $a$  on trace  $\tau$ . The time complexity of the algorithm is at most  $O(n^2)$ , where  $n$  is the length of trace  $\tau$ .

---

**Algorithm 1**  $anchor(\tau, a)$

---

```

1:  $\phi \leftarrow \langle \rangle$ ;
2:  $T \leftarrow len(\tau)$ ;
3: for  $t = T - 1 : 0$  do
4:   if  $\exists b \in \phi a, \tau(t) \not\sim b$  then
5:      $\phi \leftarrow \tau(t)\phi$ ;
6:   end if
7: end for
8: return  $len(\phi)$ ;
```

---

**Lemma 3.** For any action  $a \in A$  and trace  $\tau \in A^*$ , the function  $anchor(\tau, a)$  computes the anchor position of  $a$  on  $\tau$ .

*Proof.* For a trace  $\tau$  and an action  $a$ , algorithm  $anchor(\tau, a)$  constructs a trace  $\phi$  and returns its length. To prove that  $len(\phi)$  gives the anchor position  $k$  of  $a$  on  $\tau$ , we show both  $len(\phi) \geq k$  and  $len(\phi) \leq k$ .

$len(\phi) \geq k$ : It suffice to prove the statement by constructing a trace  $\eta$  such that  $\phi a \eta \stackrel{\varepsilon}{\equiv} \tau a$  and  $a \notin \eta$ . Let  $\eta = \tau \setminus \phi$  be the remaining subsequence of  $\tau$  after removing

the actions in  $\phi$ . We note that the ordering of actions in  $\eta$  is the same as that in  $\tau$ . For each action  $c \in \eta$ , line 5 is not executed. Hence, for all actions  $b \in \phi a$  which is originally to the right of  $c$ , we have  $b \stackrel{\varepsilon}{\sim} c$ . Therefore, action  $c$  can be swapped repeatedly to the right of action  $a$ . Repeat this process for all actions in  $\eta$ , we derive trace  $\phi a \eta$  from the original trace  $\tau a$ . Therefore  $\phi a \eta \stackrel{\varepsilon}{\equiv} \tau a$ . In addition, we note that from Definition 3, an  $\varepsilon$ -independent action pair consists of two distinctive actions, which implies  $a \not\stackrel{\varepsilon}{\sim} a$ . Hence, for each occurrence  $a \in \tau$ , line 5 is not executed, that is,  $a \notin \eta$ . Therefore, the statement holds.

$len(\phi) \leq k$ : First, we convert any trace  $\tau a$  to a trace consists of only distinctive actions. If otherwise some action  $b \in \tau a$  occurs more than once, we replace the occurrences as distinctive pseudo-actions  $b_0, b_1, \dots$ , such that each  $b_i$  inherit the same independence relation from  $b$  and any pair of these pseudo-actions is not independent. In this way, we map an arbitrary trace  $\tau a$  to a trace consists of only distinctive actions. It can be checked that this mapping is bijective. Without loss of generality, we assume that the actions in  $\tau a$  are distinctive.

We prove  $len(\phi) \leq k$  by contradiction. Suppose  $len(\phi) > k$ , then there exist traces  $\phi', \eta'$  such that (i)  $a \notin \eta'$ , (ii)  $\phi' a \eta' \stackrel{\varepsilon}{\equiv} \tau a$ , and (iii)  $len(\phi') < len(\phi)$ . From (iii), there exists an action  $c \in \phi \setminus \phi'$ . If there are multiple choices of such actions, we choose the rightmost action  $c$  in  $\phi$ . From line 4 and 5, action  $c$  is in  $\phi$  iff there exists another action  $b \in \phi$  to the right of  $c$  such that  $c \stackrel{\varepsilon}{\sim} b$ . Since we choose action  $c$  as the rightmost action in  $\phi$  that is not in  $\phi'$ , we have  $b \in \phi'$ . Originally in trace  $\tau a$ , action  $b$  is to the right of action  $c$ . As actions  $b$  and  $c$  are not  $\varepsilon$ -independent, in any equivalent trace  $\phi' a \eta' \stackrel{\varepsilon}{\equiv} \tau a$ , the relative position of them should not be changed. Hence in trace  $\phi' a \eta'$ , action  $b$  is also to the right of action  $c$ . However, since  $b \in \phi'$  and  $c \notin \phi'$ , we have action  $c$  is to the right of action  $b$  in trace  $\phi' a \eta'$ . We derive a contradiction. Therefore, if the actions in  $\tau a$  are distinctive,  $len(\phi) \leq k$ .

*Example 5.* In Example 3, we showed that  $a_0 \stackrel{\varepsilon}{\sim} a_2$  and  $a_1 \stackrel{\varepsilon}{\sim} a_2$  with  $\varepsilon = 0.1$ . It can also be checked that  $a_{\perp}$  is not  $\varepsilon$ -independent to any actions. Consider the anchor position of  $a_0$  on the trace  $\tau = a_{\perp} a_2 a_1$ . We can swap  $a_0$  ahead following the sequence  $\tau a_0 = a_{\perp} a_2 a_1 a_0 \stackrel{\varepsilon}{\equiv} a_{\perp} a_1 a_2 a_0 \stackrel{\varepsilon}{\equiv} a_{\perp} a_1 a_0 a_2$ . Since neither of  $a_{\perp}$  and  $a_1$  is independent to action  $a_0$ , it cannot be swapped ahead any further. The anchor position of  $a_0$  is 2, corresponding to its position on  $a_{\perp} a_1 a_0 a_2$ . Algorithm 1 construct a trace  $\phi = a_{\perp} a_1$ , where the length of  $\phi$  gives the correct anchor position 2.

## 6.2 Generalization of individual execution

In this section, we present an algorithm to compute a ball centered at the final state of a potential execution  $\xi$ , such that the ball contains the final states of all  $(\delta_0, \varepsilon)$ -close executions of  $\xi$ . Algorithm 2 takes inputs of an execution  $\xi = \xi_{q_0, \tau}$ , two parameters  $\delta_0, \varepsilon \geq 0$ , and a set of discrepancy functions  $\{\beta_a\}_{a \in A}$ . For each  $t \in [len(\xi)]$ , let  $a$  be the  $t$ -th action of  $\xi$  and  $\tau$  be the prefix before  $a$ . The algorithm first finds the anchor position of each action on  $\tau$ . Using the anchor positions, we can compute a sequence of representative radii  $\{\delta_t\}_{t=1}^T$  inductively. Our computation guarantees that for each  $t \in \{1, 2, \dots, T\}$ , the length  $t$  prefixes of  $\xi$  is a  $(\delta_0, \varepsilon, \delta_t)$ -representative potential execution.

---

**Algorithm 2** Generalization( $\xi, \delta_0, \varepsilon, \{\beta_a\}_{a \in A}$ )

---

```

1:  $\beta \leftarrow \max\{\beta_a\}$ ;
2:  $T \leftarrow \text{len}(\xi)$ ;
3: for  $t \in [T]$  do
4:   //let  $\tau a$  be the length  $(t+1)$  prefix of  $\text{trace}(\xi)$ 
5:    $k \leftarrow \text{anchor}(\tau, a)$ ;
6:   if  $k = t$  then
7:      $\delta_{t+1} \leftarrow \beta_a(\delta_t)$ ;
8:   else
9:      $\delta_{t+1} \leftarrow \beta_a(\delta_t) + \gamma_{t-k-1}(\varepsilon)$ ;
10:  end if
11: end for
12: return  $\mathcal{B}_{\delta_T}(\xi(T))$ ;

```

---

We will establish the correctness of Algorithm 2. First, we will show that the sequence of  $\{\delta_t\}_{t=1}^T$  is valid representative radii for the prefixes of  $\xi$ .

**Lemma 4.** *For any  $t \leq \text{len}(\xi)$ , the length  $t$  prefix of  $\xi$  is a  $(\delta_0, \varepsilon, \delta_t)$ -representative execution.*

*Proof.* For any  $t \leq \text{len}(\xi)$ , We prove the lemma by induction on  $t$ .

**Base:** For  $t = 0$ , the length 0 prefix of  $\xi$  is a single state  $q_0$ . Any  $\xi'$   $(\delta_0, \varepsilon)$ -close to  $q_0$  is also a single state  $q'_0$  with  $q'_0 \in \mathcal{B}_{\delta_0}(q_0)$ . Hence the lemma holds for  $t = 0$ .

**Induction:** Suppose the lemma holds for any prefix of  $\xi$  with length at most  $t < \text{len}(\xi)$ . We will prove the lemma holds for the length  $t+1$  prefix of  $\xi$ . Let  $q_0$  be the initial state of  $\xi$ ,  $\tau$  be the length  $t$  prefix of  $\text{trace}(\xi)$ , and  $a \in A$  is the  $(t+1)$ th action. Then  $\xi_{t+1} = \xi_{q_0, \tau a}$  is the length  $t+1$  prefix of  $\xi$ . The execution  $\xi_{t+1}$  is illustrated in Figure 6. Fix any  $\xi'$  that is  $\xi'$  is  $(\delta_0, \varepsilon)$ -close to  $\xi_{t+1}$ . From Proposition 3,  $\xi'.\text{lstate}.L = \xi_{t+1}.\text{lstate}.L$ . It suffice to prove that  $|\xi'.\text{lstate}.X - \xi_{t+1}.\text{lstate}.X| \leq \delta_{t+1}$ .

Since  $\text{trace}(\xi') \stackrel{\varepsilon}{=} \tau a$ , action  $a$  is in the sequence  $\text{trace}(\xi')$ . Partitioning  $\text{trace}(\xi')$  on the last occurrence of  $a$ , we get  $\text{trace}(\xi') = \phi a \eta$  for some  $\phi, \eta \in A^*$  with  $a \notin \eta$ . Since  $k$  is the anchor position, from Definition 7, the position of the last occurrence of  $a$  on  $\text{trace}(\xi')$  is at least  $k$ . Hence we have  $\text{len}(\phi) \geq k$  and  $\text{len}(\eta) = t - \text{len}(\phi) \leq t - k$ . We construct another potential execution  $\xi'' = \xi_{\xi'.\text{fstate}, \phi \eta a}$ . The executions  $\xi, \xi'$  and  $\xi''$  are illustrated as following.

From the inductive hypothesis, the length  $t$  prefix of  $\xi_{t+1}$ , which is the execution  $\xi_t = q_0, \tau, a_t$  in the figure is an  $(\delta_0, \varepsilon, \delta_t)$ -representative execution. We note that the length  $t$  prefix  $\xi''$  is  $(\delta_0, \varepsilon)$ -close to  $\xi_t$ . Therefore,  $|q_t.X - q_t''.X| \leq \delta_t$ . Using the discrepancy function of action  $a$ , we have

$$|q_{t+1}.X - q_{t+1}''.X| \leq \beta_a(|q_t.X - q_t''.X|) \leq \beta_a(\delta_t). \quad (10)$$

We will quantify the distance between  $\xi'$  and  $\xi''$ . There are two cases:

(i) If  $k = t$  and line 7 is executed. Then,  $\text{len}(\eta) \leq t - k = 0$ , that is,  $\eta$  is an empty string. Hence,  $\xi'$  and  $\xi''$  are indeed identical and  $q_{t+1}' = q_{t+1}''$ . Thus from (10),

$$|q_{t+1}.X - q_{t+1}'.X| = |q_{t+1}.X - q_{t+1}''.X| \leq \beta_a(\delta_t).$$



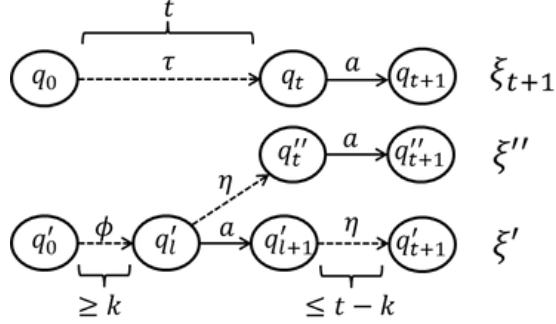


Fig. 6: Illustration of the potential executions  $\xi_{t+1}$ ,  $\xi'$ , and  $\xi''$  that is constructed by swapping action  $a$  to the back in  $\xi'$ .

Therefore if  $\delta_{t+1}$  is computed by line 7, the lemma holds for the length  $t+1$  prefix of  $\xi$ .  
(ii) Otherwise  $k < 0$  and line 9 is executed. From Lemma 1, we can bound the distance between  $\xi'$  and  $\xi''$  as

$$|q'_{t+1}.X - q''_{t+1}.X| \leq \gamma_{\text{len}(\eta)-1}(\varepsilon) \leq \gamma_{t-k-1}(\varepsilon).$$

Combining with (10), we get

$$|q_{t+1}.X - q'_{t+1}.X| \leq |q_{t+1}.X - q'_{t+1}.X| + |q'_{t+1}.X - q'_{t+1}.X| \leq \beta_a(\delta_t) + \gamma_{t-k-1}(\varepsilon).$$

Therefore if  $\delta_{t+1}$  is computed by line 9, the lemma holds for the length  $t+1$  prefix of  $\xi$ , which completes the proof.

Using this lemma, we immediately establish the soundness of the algorithm as follows.

**Theorem 1 (Soundness).** *For any execution  $\xi'$  such that  $\xi'$  is  $(\delta_0, \varepsilon)$ -close to  $\xi$ , the last state of  $\xi'$  is in the returned set  $\mathcal{B}_{\delta_T}(\xi(T))$ .*

*Proof.* From Lemma 4,  $\xi$  is a  $(\delta_0, \varepsilon, \delta_T)$ -representative potential execution. From Definition 6, for any potential execution  $\xi'$  that is  $(\delta_0, \varepsilon)$ -close to  $\xi$ , we have  $\xi'.\text{lstate} \in \mathcal{B}_{\delta_T}(\xi(T))$ .

In the following theorem, we show that the radius  $\delta_T$  can be made arbitrarily small for small enough initial radius  $\delta_0$  and the approximation parameter  $\varepsilon$ .

**Theorem 2 (Precision).** *For any  $t \leq \text{len}(\xi)$ , as  $\delta_0 \rightarrow 0$  and  $\varepsilon \rightarrow 0$ , the size of radius  $\delta_T$  goes to 0.*

*Proof.* From Proposition 4, for any  $n$ ,  $\gamma_n(\varepsilon) \rightarrow 0$  as  $\varepsilon \rightarrow 0$ . From Definition 2, for any  $\delta_t$  and discrepancy function  $\beta$ ,  $\beta(\delta_t) \rightarrow 0$  as  $\delta_t \rightarrow 0$ . Therefore, either line 6 or line 8 in Algorithm 2 is executed,  $\delta_{t+1} \rightarrow 0$  as  $\delta_t \rightarrow 0$  and  $\varepsilon \rightarrow 0$ . Iteratively applying this observation leads that,  $\delta_T \rightarrow 0$  as  $\delta_0 \rightarrow 0$  and  $\varepsilon \rightarrow 0$ . Therefore the theorem holds.

For an execution  $\xi$  of length  $n$ , the worst case time complexity of Algorithm 2 is  $O(n^3)$ , since finding the anchor position can take at most  $O(n^2)$  time. However, if all actions in  $\xi$  are mutually approximately independent, the algorithm can over-approximate the final states of  $O(n!)$  executions in the best case, which can lead to a dramatic savings in reach set computation. In the next section, we will use Algorithm 2 as a subroutine to compute an over-approximation of the reach set of a labeled transition system in a more or less standard fashion.

## 7 Reach Set Over-approximation with Partial Order Reduction

In Section 6, we presented a function *generalize()* to over-approximate the reach set of a group of  $(\delta, \epsilon)$ -close potential executions using a single potential execution. Building upon this function, we will present an algorithm to compute an over-approximation of the reach set of a labeled transition system at a time  $T$ .

### 7.1 Equivalent Action Sequences

As we defined in Section 3.1, for any set of initial states  $S \subseteq \Theta$  and time bound  $T \geq 0$ ,  $\text{Execs}(S, T)$  is the set of executions from  $S$  with length  $T$ , and  $\text{Traces}(S, T)$  is the set of action sequences taken by these executions. In concurrent systems, even with  $S$  being a singleton  $\{q\}$ , the size of action sequences  $\text{Traces}(q, T)$  can grow exponentially with  $T$ . Partial order reduction methods reduce the number of action sequences needed to be explored. These methods exploit the equivalence relation on the set of action sequences and divide  $\text{Traces}(S, T)$  into equivalence classes. Then, checking a representative action sequences suffice to cover the whole equivalence class. Here we formally define the set of representative action sequences.

**Definition 8.** Fixed any time bound  $T \geq 0$ , any equivalence relation  $\stackrel{\epsilon}{\equiv}$  on the set of length  $T$  action sequences  $A^T$  and any subset of initial state  $S \subseteq Q$ , the  $T$ -representative action sequences from  $S$  is the quotient space of the set  $\text{Traces}(S, T)$  by the equivalence relation  $\stackrel{\epsilon}{\equiv}$ .

From the above definition, for any length  $T$  execution  $\xi$  starting from  $S$ , there exists a unique action sequence  $\tau \in \mathcal{T}$ , such that  $\tau \stackrel{\epsilon}{\equiv} \text{trace}(\xi)$ . Over the past few decades, various partial order methods have been proposed to compute the representative action sequences for finite-state machines. Roughly, there are two main classes of methods that have been proposed. For each state  $q$ , the *ample/persistent set* techniques select an ample set as a subset of enabled actions from  $q$ , where the missing actions are independent to those actions in the ample set [9]. Then execution sequences that take actions from the ample sets are the representatives for equivalent classes. In contrast, the *sleep set* techniques decide which action to explore on-the-fly [30]. The technique maintains a sleep set as a set of actions that should not be explored immediately and update the set based on the independence relation along the execution. In this paper, we assume that the  $T$ -representative action sequences can be computed for any compact initial set  $S$ .

## 7.2 Reach Set Over-approximation

We propose an algorithm to compute an over-approximation of  $\text{Reach}(\Theta, T)$ . Given a labeled transition system, an approximation parameter  $\varepsilon \geq 0$ , a partition parameter  $\delta > 0$ , a time bound  $T > 0$ , and a set of discrepancy function  $\{\beta_a\}_{a \in A}$ , the algorithm computes an over-approximation of reach set at time  $T$ . In this algorithm, we first compute a  $\delta$ -cover  $Q_0$  of the initial set  $\Theta$  such that  $\mathcal{B}_\delta(Q_0) \supseteq \Theta$ . Then, in line 4, we compute the  $T$ -representative action sequences from each cover  $\mathcal{B}_\delta(q_0)$ . Then, we use the algorithm *Generalize()* to compute a ball  $\mathcal{R}'$  that over-approximate the set of states reached by a group of executions from a cover following equivalent action sequences. Then the algorithm returns a set  $\mathcal{R}$  as a union of such balls  $\mathcal{R}'$ .

---

### Algorithm 3 Reachability algorithm to over-approximate $\text{Reach}(\Theta, T)$

---

```

1:  $Q_0 \leftarrow \delta\text{-cover}(\Theta)$ ;
2:  $\mathcal{R} \leftarrow \emptyset$ ;
3: for  $q_0 \in Q_0$  do
4:    $\mathcal{T} \leftarrow \text{Traces}(\mathcal{B}_\delta(q_0)) / \stackrel{\varepsilon}{\equiv}$ ;
5:   for  $\tau \in \mathcal{T}$  do
6:      $\mathcal{R}' \leftarrow \text{Generalization}(\xi_{q_0, \tau}, \delta, \varepsilon, \{\beta_a\}_{a \in A})$ ;
7:      $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}'$ ;
8:   end for
9: end for
10: return  $\mathcal{R}$ ;

```

---

We are able to show that, for any  $T \geq 0$ , the set  $\mathcal{R}$  over-approximate the reach set of the system at time  $T$ .

**Theorem 3 (Soundness).** *For the set  $\mathcal{R}$  returned by Algorithm 3, we have  $\mathcal{R} \supseteq \text{Reach}(\Theta, T)$ .*

*Proof.* Fixed any  $\xi \in \text{Execs}(\Theta, T)$ , it suffice to show that  $\xi(T) \in \mathcal{R}$ . Since  $\xi(0) \in \Theta$  and  $Q_0$  is a  $\delta$ -cover, there exists a  $q_0 \in Q_0$  such that  $\xi(0) \in \mathcal{B}_\delta(q_0)$ . For this  $q_0$ , let  $\mathcal{T}$  be the  $T$ -presentative action sequences from  $\mathcal{B}_\delta(q_0)$  computed in line 4. From Definition 8, there must be a representative action sequence  $\tau \in \mathcal{T}$  such that  $\tau \stackrel{\varepsilon}{\equiv} \text{trace}(\xi)$ . Since  $\xi$  is  $(\delta, \varepsilon)$ -close to  $\xi_{q_0, \tau}$ , from Theorem 1, the  $\mathcal{R}'$  computed using *Generalization()* guarantees that  $\xi.lstate \in \mathcal{R}$ . Therefore the theorem holds.

We are also able to show that, we can compute the over-approximation up to arbitrary precision.

**Theorem 4 (Completeness).** *For any  $r > 0$ , there exist  $\delta, \varepsilon > 0$  such that, the set  $\mathcal{R}$  computed by Algorithm 3 satisfies  $\mathcal{R} \subseteq \mathcal{B}_r(\text{Reach}(\Theta, T))$ .*

*Proof.* Fix arbitrary  $r > 0$ . The set  $\mathcal{R}$  is a union of balls  $\mathcal{R}'$  computed in line 6. Fix any such  $\mathcal{R}'$  centered at the last state of potential execution  $\xi = \xi_{q_0, \tau}$ . It suffices to show that  $\mathcal{R}'$  satisfies  $\mathcal{R}' \subseteq \mathcal{B}_r(\text{Reach}(\Theta, T))$  for small enough  $\delta$  and  $\varepsilon$ .

Since  $\tau \in \mathcal{T}$  is a  $T$ -representative action sequence from the initial set  $\mathcal{B}_\delta(q_0)$ , from Definition 8, there exists an action sequence  $\tau' \in \text{Traces}(\mathcal{B}_\delta(q_0), T)$  such that  $\tau$  and  $\tau'$  are  $\varepsilon$ -equivalent. Hence, there is an execution  $\xi' \in \text{Execs}(\mathcal{B}_\delta(q_0), T)$  from the ball  $\mathcal{B}_\delta(q_0)$  following the action sequence  $\tau'$ . By the definition of reach set, we have  $\xi'.\text{lstate} \in \text{Reach}(\Theta, T)$ . On the other hand,  $\xi'$  is  $(\delta, \varepsilon)$ -close to the potential execution  $\xi$ . From Theorem 1,  $\xi'.\text{lstate}$  is in the ball  $\mathcal{R}'$ . That is, the ball  $\mathcal{R}'$  and the reach set  $\text{Reach}(\Theta, T)$  intersect at the state  $\xi'.\text{lstate}$ .

From Theorem 2, the radius of  $\mathcal{R}'$  can be made arbitrarily small as  $\delta$  and  $\varepsilon$  go to 0. We chose small enough  $\delta$  and  $\varepsilon$ , such that the radius of  $\mathcal{R}'$  is less than  $r/2$ . Therefore, the ball  $\mathcal{R}'$  is contained in the radius  $r$  ball of the reach set  $\mathcal{B}_r(\text{Reach}(\Theta, T))$ .

In this section, we proposed an algorithm to over-approximate the reach set of label transition systems. In addition, we showed that the computation can be made arbitrarily precise. We will apply our method to verification of a linear transition system and a temperature control system.

## 8 Case Studies

### 8.1 Linear transition systems

We will first study the linear transition systems as presented in Example 1. In Example 3, we constructed an instance of the system with  $N = 3$ , and showed that  $a_0 \stackrel{\varepsilon}{\sim} a_2$  and  $a_1 \stackrel{\varepsilon}{\sim} a_2$ . The instance of the system has 3 continuous variables and 3 actions  $a_0, a_1, a_2$ . We also showed that  $a_0 \stackrel{\varepsilon}{\sim} a_2$  and  $a_1 \stackrel{\varepsilon}{\sim} a_2$  with  $\varepsilon = 0.1$ . In Example 4 we further presented an execution and its  $(\delta, \varepsilon)$ -close executions.

For this system, we want to prove if the continuous states can converge to a box  $[-0.5, 0.5]^3$  in 3 rounds. We illustrate an execution  $\xi = \xi_{q_0, \tau}$  with  $q_0.x = [2.5, -3.5, 1.2]$  and  $\tau = a_0 a_2 a_1 a_\perp a_2 a_1 a_0 a_\perp a_1 a_0 a_2 a_\perp$  in Figure 7 (blue curve). Using Algorithm 2, we compute a tube around the execution to over-approximate all  $(\delta, \varepsilon)$ -close potential executions (green curves). To give an example of  $\delta, \varepsilon$ -close potential executions to  $\xi$ , we compute a potential execution  $\xi' = \xi_{q'_0, \tau}$  with  $q'_0.x = [2.3, -3.2, 1]$  and  $\tau' = a_0 a_1 a_2 a_\perp a_1 a_0 a_2 a_\perp a_2 a_1 a_0 a_\perp$  (red curve). The result validates that  $\xi'$  lies in the tube computed using our technique.

### 8.2 Room heating problem

We present a building heating system in Fig. 8. The building has  $N$  rooms each with a heater. For  $i \in [N]$ ,  $x_i \in \mathbb{R}$  is the temperature of room  $i$  and  $m_i \in \{0, 1\}$  captures the off/on state of the heater in the room. The building measures the temperature of rooms periodically every  $T$  seconds and save the measurements to  $y_i$ . Based on the measurement  $y_i$ , each room takes action  $a_i$  to decide whether to turn on or turn off its heater. The boolean variable  $d_i$  indicates whether room  $i$  has made a decision. These decisions are made asynchronously among the rooms with a small delay  $h$ . For this system, we want to check whether the temperature of the room remains in an appropriate range.

For  $i \in [N]$ , actions  $on_i, off_i$  capture the decision making process of room  $i$  on whether or not to turn on the heater. During the process, time elapses for a (short)

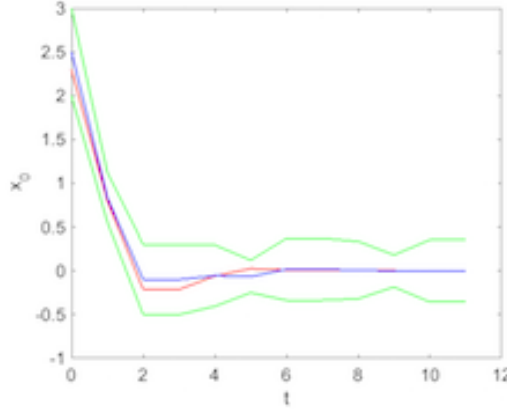


Fig. 7: A tube around an execution of a linear transition system. The blue curve is the execution  $\xi$ . The green curves illustrate a tube around  $\xi$ . The red curve is a  $(\delta, \varepsilon)$ -close potential execution with  $\delta = 0.5$  and  $\varepsilon = 0.1$ .

<b>automata</b> Roomheating( $N : \text{Nat}$ )		
2	<b>variables</b>	
	$x : \text{Real}^N$ <b>initially</b> $x[i] := 60$ ;	
4	$y : \text{Real}^N$ <b>initially</b> $y[i] := 60$ ;	
	$d : \text{Bool}^N$ <b>initially</b> $d := \text{false}^N$ ;	
6	$m : \text{Bool}^N$ <b>initially</b> $m := \text{false}^N$ ;	
8	<b>transitions</b>	
	on <sub>i</sub> , <b>for</b> $i \in [N]$	
10	<b>pre</b> $!d_i \wedge y_i \leq 72$	
	<b>eff</b> $x := W(h)x + b(h) + C(h)m$ ;	
12	$d_i := \text{true} \wedge m_i := \text{true}$ ;	
	off <sub>i</sub> , <b>for</b> $i \in [N]$	
	<b>pre</b> $!d_i \wedge y_i \geq 68$	
	<b>eff</b> $x := W(h)x + b(h) + C(h)m$ ;	
	$d_i := \text{true} \wedge m_i := \text{false}$ ;	
	flow	
	<b>pre</b> $\bigwedge_{i \in [N]} d_i$	
	<b>eff</b> $x := W(T)x + b(T) + C(T)m$ ;	
	$d_i := \text{false}$ <b>for each</b> $i \in [N]$ ;	
	$y := x$ ;	

Fig. 8: Transition system of room heating.

period  $h$ , which leads to an update of the temperature as an affine function of current temperature  $x$  and the heaters state  $m$ . The affine function is derived from the thermal equations presented in [16]. In this section, we use an instance of the system with the following matrices:

$$W(h) = \begin{bmatrix} 0.96 & 0.01 & 0.01 \\ 0.02 & 0.97 & 0.01 \\ 0 & 0.01 & 0.97 \end{bmatrix}, b(h) = \begin{bmatrix} 1.2 \\ 0 \\ 1.2 \end{bmatrix}, C(h) = \begin{bmatrix} 0.4 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.4 \end{bmatrix}. \quad (11)$$

After a room controller makes a decision (*on<sub>i</sub>* or *off<sub>i</sub>* transition occurs), the variable  $d_i$  to *true*. After all rooms make their decisions, action *flow* captures the time elapse for a (longer) period  $T$  which also updates the measured values  $y$ . We use an instance of

this step with the following matrices:

$$W(T) = \begin{bmatrix} 0.18 & 0.11 & 0.14 \\ 0.18 & 0.25 & 0.17 \\ 0.09 & 0.13 & 0.28 \end{bmatrix}, b(T) = \begin{bmatrix} 34.2 \\ 24 \\ 30 \end{bmatrix}, C(T) = \begin{bmatrix} 11.4 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 10 \end{bmatrix}. \quad (12)$$

For each  $i \in [N]$  and  $a \in \{on_i, off_i\}$ , we will derive the discrepancy function for action  $a$ . For any  $q, q'$  with  $q.L = q'.L$ ,

$$\begin{aligned} & |a_i(q).x - a_i(q').x| \\ &= |W(h)q.x + b(h) + C(h)q.m - W(h)q'.x - b(h) - C(h)q'.m| \\ &\leq |W(h)||q.x - q'.x| \end{aligned}$$

We note that  $|W(h)| = 0.99$ . Hence, we can define  $\beta_a(|q.x - q'.x|) = 0.99|q.x - q'.x|$  as the discrepancy functions of each  $a \in \{on_i, off_i\}_{i \in [3]}$ . Similarly, we derived that  $\beta_{flow}(|q.x - q'.x|) = 0.52|q.x - q'.x|$ .

For any  $i, j \in [3]$  with  $i \neq j$ ,  $a \in \{on_i, off_i\}$  and  $b \in \{off_j, off_j\}$ , we can prove  $a \stackrel{\varepsilon}{\sim} b$  with  $\varepsilon = 0.6$ . Notice that,  $a_i(q).x = W(h)q.x + b(h) + C(h)q.m = a_j(q).x$  are identical, but  $a_i(q).m$  and  $a_j(q).m$  could be different.

$$\begin{aligned} & |a_i a_j(q).x - a_j a_i(q).x| \\ &= |W(h)a_j(q).x + b(h) + C(h)a_j(q).m - W(h)a_i(q).x - b(h) - C(h)a_i(q).m| \\ &= |C(h)a_j(q).m - C(h)a_i(q).m| \leq |C(h)||a_j(q).m - a_i(q).m| \end{aligned}$$

We note that  $|C(h)| = 0.4$ . We will give an upper bound on  $|a_j(q).m - a_i(q).m|$ . Notice that  $a_i(q).m$  and  $q.m$  can only differ in one bit ( $m_i$ ). Similarly,  $a_j(q).m$  and  $q.m$  can only differ in one bit ( $m_j$ ). Hence  $a_i(q).m$  and  $a_j(q).m$  can be differ in at most two bits, and  $|a_i(q).m - a_j(q).m| \leq |[1, 1, 0]| = 1.41$ . Therefore,

$$|a_i a_j(q).x - a_j a_i(q).x| \leq 0.4 * 1.41 \leq 0.6.$$

Thus for any pair of rooms, the on/off decisions are  $\varepsilon$ -approximately independent with  $\varepsilon = 0.6$ . For a round, where each room makes a decision once in arbitrary order, there are in total  $3! = 6$   $\varepsilon$ -equivalent action sequences.

We present an execution  $\xi$  in Figure 9 as the blue curve, which ran for 8 rounds. There are a number of 1.6 million ( $6^8$ )  $(\delta, \varepsilon)$ -close potential executions of  $\xi$  with  $\varepsilon = 0.6$  and  $\delta = 2$ . We illustrate a tube computed by our technique in green, which is proved to contain the final state of all these  $(\delta, \varepsilon)$ -close potential executions.

## 9 Conclusion

In this paper, we propose a partial order reduction technique for infinite state transition systems. We proposed the notion of  $\varepsilon$ -independent actions as an extension of the conventional notion of independent actions, such that the resulting states are within  $\varepsilon$  distance regardless of the order of executing a pair of  $\varepsilon$ -independent actions. The conventional notion of independent actions is indeed a special case of  $\varepsilon$ -independent actions

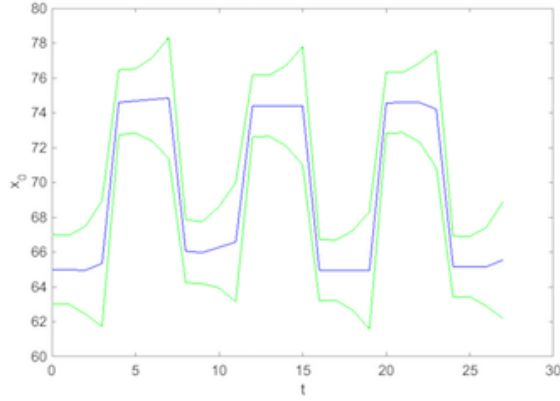


Fig. 9: A tube around an execution of a room heating problem. The blue curve is the execution  $\xi$ . The green curves illustrate a tube around  $\xi$  over-approximating  $(\delta, \varepsilon)$ -close potential execution with  $\delta = 2$  and  $\varepsilon = 0.6$ .

with  $\varepsilon = 0$ . With this  $\varepsilon$ -independence relation, we are able to define an  $\varepsilon$ -equivalence relation among traces by swapping consecutive  $\varepsilon$ -independent actions.

We derive the distance between two executions after inserting a single action at different positions (Lemma 1). We observe that, the distance between executions depends on the difference in the inserting positions. Based on this result, we proposed an algorithm to compute over-approximated reach set of all executions with  $\varepsilon$ -equivalent traces. In  $O(n^3)$  time, the algorithm can over-approximate executions with at most  $O(n!)$  distinct traces. We applied the algorithm to verify properties of a linear transition system and a heater control system.

The results of this paper suggest several interesting future directions. In Definition 3,  $\varepsilon$ -independent actions are required to be approximately commutative globally. That is, from *any* state  $q$ , the resulting states after executing  $\varepsilon$ -independent actions in any order are close to each other. Such definition ignores action pairs that are only approximately commutative in a subset of states. A logical next step is to extend the notion of  $\varepsilon$ -independent actions to capture locally independent action pairs. An orthogonal future direction is to verify temporal properties using our reachability analysis. One possible approach for achieving this is by combining our reach set over-approximation algorithm with temporal precedence checking algorithm, such as the one proposed in [12].

## References

1. Abdulla, P., Aronis, S., Jonsson, B., Sagonas, K.: Optimal dynamic partial order reduction. In: ACM SIGPLAN Notices. vol. 49, pp. 373–384. ACM (2014)
2. Alur, R., Brayton, R.K., Henzinger, T.A., Qadeer, S., Rajamani, S.K.: Partial-order reduction in symbolic state space exploration. In: International Conference on Computer Aided Verification. pp. 340–351. Springer (1997)

3. Baier, C., Größer, M., Ciesinski, F.: Partial order reduction for probabilistic systems. In: QEST. vol. 4, pp. 230–239 (2004)
4. Baier, C., Katoen, J.P., Larsen, K.G.: Principles of model checking. MIT press (2008)
5. Blondel, V., Hendrickx, J.M., Olshevsky, A., Tsitsiklis, J., et al.: Convergence in multi-agent coordination, consensus, and flocking. In: IEEE Conference on Decision and Control. vol. 44, p. 2996. IEEE; 1998 (2005)
6. Chaudhuri, S., Gulwani, S., Lubliner, R.: Continuity and robustness of programs. Communications of the ACM 55(8), 107–115 (2012)
7. Clarke, E., Jha, S., Marrero, W.: Partial order reductions for security protocol verification. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 503–518. Springer (2000)
8. Clarke, E.M., Grumberg, O., Minea, M., Peled, D.: State space reduction using partial order techniques. International Journal on Software Tools for Technology Transfer 2(3), 279–287 (1999)
9. Clarke, E.M., Grumberg, O., Peled, D.: Model checking. MIT press (1999)
10. Donzé, A., Maler, O.: Systematic simulation using sensitivity analysis. In: Hybrid Systems: Computation and Control, pp. 174–189. Springer (2007)
11. Duggirala, P., Mitra, S., Viswanathan, M.: Verification of annotated models from executions. In: International Conference on Embedded Software (2013)
12. Duggirala, P.S., Wang, L., Mitra, S., Viswanathan, M., Munoz, C.: Temporal precedence checking for switched models and its application to a parallel landing protocol. In: International Symposium on Formal Methods. pp. 215–229. Springer (2014)
13. Duggirala, P., Mitra, S., Viswanathan, M., Potok, M.: C2E2: A verification tool for stateflow models. In: Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 9035, pp. 68–82. Springer Berlin Heidelberg (2015)
14. Fan, C., Mitra, S.: Bounded verification with on-the-fly discrepancy computation. In: International Symposium on Automated Technology for Verification and Analysis (ATVA), 2015 (2015), <http://arxiv.org/abs/1502.01801>
15. Fang, L., Antsaklis, P.J.: Information consensus of asynchronous discrete-time multi-agent systems. In: Proceedings of the 2005, American Control Conference, 2005. pp. 1883–1888. IEEE (2005)
16. Fehnker, A., Ivančić, F.: Benchmarks for hybrid systems verification. In: International Workshop on Hybrid Systems: Computation and Control. pp. 326–341. Springer (2004)
17. Flanagan, C., Godefroid, P.: Dynamic partial-order reduction for model checking software. In: ACM Sigplan Notices. vol. 40, pp. 110–121. ACM (2005)
18. Godefroid, P., van Leeuwen, J., Hartmanis, J., Goos, G., Wolper, P.: Partial-order methods for the verification of concurrent systems: an approach to the state-explosion problem, vol. 1032. Springer Heidelberg (1996)
19. Huang, Z., Mitra, S.: Proofs from simulations and modular annotations. In: Proceedings of the 17th international conference on Hybrid systems: computation and control. pp. 183–192. ACM (2014)
20. Huang, Z., Mitra, S., Dullerud, G.: Differentially private iterative synchronous consensus. In: Proceedings of the 2012 ACM workshop on Privacy in the electronic society. pp. 81–90. WPES '12, ACM, New York, NY, USA (2012), <http://doi.acm.org/10.1145/2381966.2381978>
21. Majumdar, R., Saha, I.: Symbolic robustness analysis. In: Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE. pp. 355–363. IEEE (2009)
22. Mitra, D.: An asynchronous distributed algorithm for power control in cellular radio systems. In: Wireless and Mobile Communications, pp. 177–186. Springer (1994)



23. Mitra, S., Chandy, K.M.: A formalized theory for verifying stability and convergence of automata in pvs. In: International Conference on Theorem Proving in Higher Order Logics. pp. 230–245. Springer (2008)
24. Olfati-Saber, R., Fax, J.A., Murray, R.M.: Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE* 95(1), 215–233 (2007)
25. Peled, D.: Verification for robust specification. In: International Conference on Theorem Proving in Higher Order Logics. pp. 231–241. Springer (1997)
26. Peled, D.: Ten years of partial order reduction. In: International Conference on Computer Aided Verification. pp. 17–28. Springer (1998)
27. Rhee, I.K., Lee, J., Kim, J., Serpedin, E., Wu, Y.C.: Clock synchronization in wireless sensor networks: An overview. *Sensors* 9(1), 56–85 (2009)
28. Samanta, R., Deshmukh, J.V., Chaudhuri, S.: Robustness analysis of networked systems. In: International Workshop on Verification, Model Checking, and Abstract Interpretation. pp. 229–247. Springer (2013)
29. Welch, J.L., Lynch, N.: A new fault-tolerant algorithm for clock synchronization. *Information and computation* 77(1), 1–36 (1988)
30. Yang, Y., Chen, X., Gopalakrishnan, G., Kirby, R.M.: Efficient stateful dynamic partial order reduction. In: International SPIN Workshop on Model Checking of Software. pp. 288–305. Springer (2008)